# Developing improved numerical algorithms for solving highly complex partial differential equations

**M. M. Sadiq Aziz Hussein**

**Third Rusafa Education Directorate**

Aziz7651@jamil.com

**Abstract**

Partial differential equations (PDEs) are a fundamental mathematical tool for modeling many physical and engineering phenomena. However, the development and increasing complexity of scientific models have led to the emergence of highly complex PDEs characterized by nonlinearity, multiple spatial and time scales, variable coefficients, and complex boundary and geometric conditions. This complexity makes it difficult to obtain closed analytical solutions, necessitating reliance on numerical algorithms as the primary practical option. However, many traditional numerical algorithms suffer from limitations in accuracy and stability, as well as high computational costs, when applied to this type of equation. This research aims to develop improved numerical algorithms capable of solving highly complex PDEs more efficiently, while maintaining high levels of accuracy and numerical stability, and reducing computation time and resource consumption. The research employs a methodological framework that combines theoretical analysis and numerical experimentation, through studying the properties of high-complexity equations, reviewing spatial and time differentiation methods, and analyzing the concepts of stability, convergence, and error control. The research also explores advanced strategies such as h/p/hp-adaptivity, the use of presets and multigrid methods, and the application of parallel computing to enhance computational performance. The proposed algorithms are tested using a set of standard and applied problems representing different patterns of numerical complexity, and their performance is evaluated according to the criteria of accuracy, stability, convergence order, and computational efficiency.

**Keywords: Partial differential equations, numerical algorithms, numerical stability, numerical convergence, network adaptation, error control, multigrid, parallel computing.**

## 1.1 Introduction

Partial Differential Equations (PDEs) constitute one of the fundamental pillars of mathematical modeling for natural and engineering phenomena. They are widely used to describe complex physical processes such as heat transfer, wave propagation, fluid dynamics, chemical reactions, mass transport, and electromagnetic phenomena. These equations enable researchers to understand the spatial and temporal

behavior of continuous systems and to predict their evolution under various conditions. With the advancement of science and technology, mathematical models have become increasingly complex and precise, leading to the emergence of partial differential equations characterized by strong nonlinearity, multiple spatial and temporal scales, variable coefficients, as well as complex boundary and geometric conditions. This growing complexity makes it difficult—if not impossible in many cases—to obtain closed-form analytical solutions, thereby necessitating reliance on numerical solutions as a primary and practical alternative. Despite significant progress in numerical analysis, many traditional numerical algorithms still suffer from fundamental limitations when applied to highly complex PDEs, including weak numerical stability, accumulation of approximation errors, and the need for extremely fine computational meshes, which result in high computational cost and excessive consumption of computational resources. These challenges become more pronounced in nonlinear and multiscale problems, where maintaining solution accuracy requires very small time steps or dense spatial discretizations, leading to long and computationally expensive simulations, particularly in large-scale scientific and engineering applications. Moreover, the rapid development of high-performance computing and the increasing volume of data generated by numerical simulations impose new requirements on numerical algorithms in terms of efficiency, parallel scalability, and optimal utilization of available resources. Accordingly, there is an urgent need to develop improved numerical algorithms capable of efficiently solving highly complex partial differential equations while maintaining high levels of accuracy and numerical stability. fields.

### 2.1 Research Problem

The research problem lies in the limited efficiency of traditional numerical algorithms used to solve highly complex partial differential equations. These algorithms often encounter difficulties in maintaining numerical stability and the required level of accuracy when applied to nonlinear or multiscale problems, in addition to high computational time and memory consumption. Accordingly, the main research question can be formulated as follows: How can improved numerical algorithms be developed to solve highly complex partial differential equations with high accuracy and stability while reducing computational cost?

### 3.1 Significance of the Research

The significance of this research stems from several interrelated aspects. From a scientific perspective, it contributes to the development of theoretical knowledge in the numerical analysis of partial differential equations and proposes more efficient numerical methods and algorithms. From an applied perspective, improving numerical algorithms directly enhances the quality of computational simulations in

engineering, physics, and environmental applications, thereby supporting more accurate and reliable decision-making. From a computational standpoint, the research aims to reduce computational time and resource consumption, which is critically important given the growing reliance on large-scale numerical simulations. In addition, the research provides a methodological framework that can be employed in the development of more advanced numerical applications in the future.

**4.1 Research Objectives** The main objective of this research is to develop improved numerical algorithms for solving highly complex partial differential equations. This general objective is supported by several specific objectives, including:

1- Analyzing the numerical challenges associated with solving complex partial differential equations.
2- Studying and evaluating traditional numerical algorithms in terms of accuracy, stability, and efficiency.
3- Proposing improved numerical algorithms based on more advanced differentiation and integration techniques.
4- Testing the efficiency of the proposed algorithms through their application to benchmark problems.
5- Comparing the results of the developed algorithms with traditional methods and highlighting areas of improvement.

**5.1 Research Hypotheses**

This research is based on a set of scientific hypotheses, the most important of which are:

1- Developing improved numerical algorithms leads to enhanced numerical accuracy in solving highly complex partial differential equations.
2- The use of advanced numerical methods contributes to improving numerical stability and reducing undesirable oscillations in the solution.
3- Improved numerical algorithms are capable of reducing computational time compared to traditional algorithms while achieving the same level of accuracy.

**6.1 Scope of the Research**

The scope of this research is defined thematically by focusing on the development of numerical algorithms for solving highly complex partial differential equations, without addressing analytical solutions. Spatially, the study is limited to computational applications within numerical simulation environments. Temporally, the research is confined to the time period allocated for conducting the study and performing simulation experiments. The study is further restricted to selected models of partial differential equations with significant practical relevance.

**7.1 Key Terminology**

- Partial Differential Equations (PDEs): Mathematical equations that involve partial derivatives of functions depending on more than one independent variable.
- Numerical Algorithms: Sequential computational procedures used to obtain approximate solutions to mathematical problems.
- Numerical Stability: The ability of an algorithm to produce bounded solutions under perturbations in data or computational steps.
- Numerical Accuracy: The degree to which a numerical solution approximates the exact or reference solution.
- Computational Cost: The amount of time and computational resources required to execute an algorithm.

## Chapter Two: Theoretical Framework

Classification of Highly Complex Partial Differential Equations and Numerical Challenges

Partial Differential Equations (PDEs) constitute a fundamental mathematical tool for describing many physical and engineering phenomena. However, the increasing complexity of realistic models has led to the emergence of a class of equations known as highly complex partial differential equations, which possess characteristics that make their numerical solution more challenging than that of conventional equations. Classifying these equations and understanding the associated numerical challenges represent a crucial step toward developing improved numerical algorithms capable of handling them efficiently (Al-Bayati, 2017).

### 2.1 Classification of Highly Complex Partial Differential Equations

Highly complex PDEs can be classified according to several criteria, most notably the nature of the equation, the behavior of the solution, and the structure of the domain and coefficients.

Nonlinear Partial Differential Equations (Nonlinear PDEs)

Nonlinear PDEs are among the most complex types of partial differential equations, as the variables or their derivatives are not related through linear relationships. This form of nonlinearity gives rise to complex physical phenomena such as strong interactions, solution instabilities, the formation of shocks, or chaotic patterns. Common examples include the Burgers equation and the Navier–Stokes equations. Nonlinearity poses a major challenge in numerical analysis, as it requires specialized algorithms to ensure convergence and numerical stability.

Multiscale Partial Differential Equations (Multiscale PDEs)

These equations are characterized by the presence of significant differences in spatial or temporal scales within the same problem, such as the coexistence of fast and slow processes. Multiscale PDEs commonly arise in the modeling of composite materials, heat transfer in heterogeneous media, and fluid flow in porous structures. The presence of multiple scales makes it difficult to accurately represent all phenomena using a uniform computational mesh, thereby significantly increasing the computational burden.

Stiff Partial Differential Equations (Stiff PDEs)

Stiff PDEs exhibit components that vary very rapidly compared to other, slower components, imposing severe restrictions on the time step size in numerical simulations. Such equations frequently occur in reaction–diffusion problems and in chemical or biological models. Addressing stiffness typically requires the use of implicit or semi-implicit time-integration schemes to ensure numerical stability.

**Equations with Variable or Irregular Coefficients**

In many real-world applications, the coefficients of partial differential equations are neither constant nor smooth; instead, they may vary spatially or temporally, or even be irregular. This variability introduces additional difficulties in numerical approximation, as sharp gradients or discontinuities may appear in the solution, necessitating fine spatial resolution or specialized numerical stabilization techniques (Al-Jubouri, 2015, p. 102).

**Equations with Complex Boundary Conditions and Geometries**

The level of complexity increases further when PDEs are defined on irregular or geometrically complex domains with diverse boundary conditions, such as Dirichlet, Neumann, or mixed boundary conditions. These factors directly influence the choice of spatial discretization methods and the construction of appropriate computational meshes.

## 2.2 Numerical Challenges Associated with Highly Complex PDEs

The numerical solution of highly complex partial differential equations is associated with several fundamental challenges, including:

Numerical Stability Maintaining numerical stability is one of the most critical challenges, as inappropriate choices of time step size or spatial discretization can lead to divergent or non-physical solutions.

**Accuracy and Error Control**

Accurately capturing complex physical phenomena requires high numerical precision; however, increasing accuracy often comes at the expense of higher computational cost. This highlights the importance of error estimation and error control techniques.

**High Computational Cost**

The large number of degrees of freedom involved in complex PDE problems results in substantial increases in computational time and memory consumption, limiting the applicability of traditional algorithms to large-scale problems.

**Treatment of Nonlinearity**

Nonlinear equations require iterative solution techniques, such as Newton-type methods, which may suffer from slow convergence or even failure in certain cases.

Parallel ScalabilityWith the advancement of high-performance computing, numerical algorithms must be parallelizable and capable of efficiently exploiting multi-processor architectures to achieve scalability and computational efficiency.

## 2.3 Review of Spatial and Temporal Discretization Methods

Spatial and temporal discretization methods constitute the core components of numerical solutions for partial differential equations, as they aim to transform continuous equations into algebraic systems that can be solved using computers. The selection of an appropriate method depends on the nature of the equation, its level of complexity, and the requirements for accuracy, stability, and computational efficiency. In this context, numerical solution methods can be broadly divided into two main categories: spatial discretization methods and temporal discretization methods (Hussein, 2014, p. 78).

### 3.1.2 Spatial Discretization Methods

Spatial discretization aims to approximate the spatial derivatives in partial differential equations and constitutes the primary factor in determining the spatial accuracy of the numerical solution.

1-Finite Difference Method (FDM) The finite difference method is one of the oldest and simplest spatial discretization techniques. It is based on approximating derivatives using differences between function values at regularly spaced grid points. This method is characterized by its ease of implementation and low computational complexity. However, its applicability becomes limited when dealing with complex geometries or irregular boundary conditions. Moreover, improving accuracy requires reducing the mesh size, which leads to a significant increase in computational cost.

2-Finite Volume Method (FVM) The finite volume method is based on the principle of conservation of physical quantities. The governing equations are integrated over small control volumes, and fluxes are computed across the boundaries of these volumes. This method is widely used in fluid dynamics and heat transfer problems due to its strong conservation properties, even in the presence of shocks or sharp gradients. Nevertheless, the design of stable and high-order accurate flux schemes remains a major challenge, particularly in multidimensional problems.

3-Finite Element Method (FEM) The finite element method is among the most flexible and powerful spatial discretization approaches. It relies on partitioning the domain into small elements and using local approximation functions to represent the solution. FEM is especially effective in handling complex geometries and variable coefficients and allows the use of higher-order elements to enhance accuracy. However, its implementation generally requires greater effort in mathematical formulation and programming compared to finite difference and finite volume methods.

4-Discontinuous Galerkin Method (DG) The discontinuous Galerkin method represents an extension of the finite element method in which solution continuity between elements is not enforced. This approach provides high accuracy, excellent parallel scalability, and strong flexibility for mesh adaptivity. However, it suffers from an increased number of degrees of freedom and higher computational cost, necessitating the use of additional optimization techniques (Al-Rawi, 2016, p. 119).

### 3.2.2 Temporal Discretization Methods

Temporal discretization aims to approximate time derivatives and ensure accurate tracking of the solution evolution over time. It has a direct impact on numerical stability.

1- Explicit Methods

Explicit methods, such as the explicit Euler method and Runge–Kutta schemes, compute the solution at the new time level using known values from the current time level. These methods are simple and easy to implement; however, they are subject to severe restrictions on the time step due to stability requirements, most notably the Courant–Friedrichs–Lewy (CFL) condition. As a result, they are generally unsuitable for stiff or multiscale problems.

2-Implicit Methods In implicit methods, such as the implicit Euler method and backward differentiation formulas (BDF), the solution at the new time level depends on unknown values, requiring the solution of a system of equations at each time step. These methods exhibit superior stability properties, particularly

for stiff problems, but incur higher computational costs due to the need to solve linear or nonlinear systems.

3-Semi-Implicit Methods Semi-implicit methods aim to balance the simplicity of explicit schemes with the stability of implicit schemes by treating certain terms implicitly and others explicitly. These methods are efficiently used in advection–diffusion problems and help reduce computational time while maintaining acceptable stability levels (Al-Samarrai, 2012, p. 33).

4-Implicit–Explicit (IMEX) Methods IMEX methods combine explicit and implicit approaches by treating stiff terms implicitly and non-stiff terms explicitly. These methods are considered advanced and effective techniques for solving highly complex partial differential equations, particularly those involving both fast and slow dynamics.

### 3.2 Stability, Convergence, and Error Control

The concepts of stability, convergence, and error control constitute fundamental pillars in the numerical analysis of partial differential equations, as they represent the primary criteria for assessing the efficiency and reliability of numerical algorithms. The success of a numerical method is not limited to its ability to produce an approximate solution; it also requires ensuring that the solution is stable, converges toward the exact solution, and that the associated error remains within acceptable and controllable bounds.

#### 3.2.1 Numerical Stability

Numerical stability refers to the behavior of a numerical solution in response to small perturbations in initial data, boundary conditions, or computational operations. A numerical method is considered stable if such perturbations do not lead to unbounded growth in the solution as time advances or as the number of computational steps increases. Stability is a crucial requirement, particularly in time-dependent problems, where inappropriate choices of time step size or spatial discretization may result in divergent or non-physical solutions. Stability requirements vary depending on the type of partial differential equation and the discretization method employed. In explicit methods, stability is often governed by the Courant–Friedrichs–Lewy (CFL) condition, which relates the time step to the spatial mesh size and the propagation speed of the physical phenomenon. Violation of this condition leads to instability. Implicit methods, on the other hand, generally exhibit higher stability and may be unconditionally stable, albeit at the cost of increased computational effort.

### 3.2.2 Numerical Convergence

Numerical convergence is defined as the ability of a numerical solution to approach the exact solution of a partial differential equation as the time step and spatial mesh size are reduced. Convergence is a key criterion for evaluating the accuracy and theoretical validity of a numerical method. The rate of convergence is typically related to the order of the numerical scheme used for spatial and temporal discretization. Convergence is closely linked to both stability and consistency. According to the Lax Equivalence Theorem, a linear numerical method is convergent if and only if it is both consistent and stable. This theorem highlights the essential role of stability in ensuring convergence. For nonlinear equations, proving convergence becomes more complex and often relies on numerical experiments and approximate analytical arguments rather than rigorous mathematical proofs (Abdullah, 2018, p. 91).

### 3.2.3 Numerical Error and Its Types

Numerical error in solving partial differential equations arises from several sources, most notably spatial and temporal discretization errors, round-off errors associated with floating-point arithmetic, and modeling errors. Numerical errors can be classified into two main categories: a priori error, which is estimated theoretically before computations are performed, and a posteriori error, which is evaluated after obtaining the numerical solution. Understanding the sources and nature of numerical error is a fundamental step toward developing more efficient numerical algorithms, as it enables the identification of regions or time intervals that require higher computational accuracy (Brenner & Scott, 2008, p. 145).

### 3.4.2 Error Control

Error control refers to the set of techniques used to reduce numerical error or to keep it within prescribed bounds. One of the most prominent error control strategies is adaptive mesh refinement (AMR), in which the spatial mesh is refined in regions where high error levels are detected. In addition, temporal adaptivity techniques are employed to adjust the time-step size according to the behavior of the solution. A posteriori error estimators play a crucial role in this context, as they provide a practical measure of the quality of the numerical solution and are used to guide decisions regarding spatial or temporal adaptivity. This approach improves computational efficiency by concentrating computational resources in the most sensitive regions rather than distributing them uniformly across the domain.

### 3.5.2 Preconditioning, Multigrid, and Parallelization

The efficient solution of large algebraic systems arising from the numerical discretization of partial differential equations (PDEs) represents one of the major challenges in modern scientific computing. As the accuracy of spatial and temporal discretization increases, the number of degrees of freedom grows significantly, leading to very large linear or nonlinear systems that are difficult to solve using conventional methods. In this context, the use of preconditioning techniques, multigrid methods, and parallel computing has become essential for improving computational performance and accelerating the solution process.

1- Preconditioning

Preconditioning refers to techniques used to improve the properties of the algebraic system resulting from numerical discretization, with the goal of accelerating the convergence of iterative solvers. When solving large linear systems using iterative methods such as the Conjugate Gradient (CG) method or Krylov subspace methods, convergence is often slow due to the ill-conditioning of the system matrix. Preconditioners transform the original system into an equivalent form that is more favorable for iterative solution (Cockburn et al., 2000).

Preconditioners can be classified into several types, including simple preconditioners such as the Jacobi and Gauss–Seidel preconditioners, as well as more advanced approaches based on incomplete factorizations, such as Incomplete LU (ILU) or Incomplete Cholesky factorizations. These preconditioners are effective in reducing the number of iterations required to reach convergence; however, the choice of an appropriate preconditioner depends on the nature of the problem and the structure of the system matrix.

**2- Multigrid Methods**

Multigrid methods are among the most efficient techniques for solving the algebraic systems arising from PDE discretization. They are based on the idea of treating errors of different scales on grids with varying levels of resolution. While traditional iterative methods efficiently reduce high-frequency errors, they are less effective at eliminating low-frequency errors, which multigrid methods address by operating on coarser grids. Multigrid algorithms involve transferring information between fine and coarse grids through restriction and prolongation operators, in addition to applying smoothing procedures such as Jacobi or Gauss–Seidel iterations.

Multigrid methods are characterized by their rapid convergence rates, often achieving solutions in a number of iterations that scales linearly with the number of degrees of freedom, making them particularly well suited for large-scale problems. Furthermore, multigrid techniques are not only used as standalone solvers but also serve as highly effective preconditioners within Krylov methods, such as using multigrid as a preconditioner for GMRES, thereby combining the advantages of both approaches.

3- Parallelization and High-Performance Computing

With the rapid advancement of multi-core computer architectures and parallel processing systems, parallel scalability has become a fundamental requirement for modern numerical algorithms. Parallelization aims to distribute computational tasks across multiple processors operating simultaneously, thereby reducing overall execution time and improving resource utilization. Spatial discretization methods such as the finite element method and the discontinuous Galerkin method are naturally well suited for parallel implementation due to their reliance on local operations within individual elements.

Moreover, multigrid methods and advanced preconditioners can be adapted to parallel computing environments using programming models such as MPI or OpenMP. Despite these advantages, parallel computing introduces challenges related to memory management, inter-processor communication costs, and load balancing, all of which must be carefully addressed to achieve optimal performance (Evans, 2010, p. 210).

## Chapter Three

## Methodology for Algorithm Development

The methodology for algorithm development represents the practical framework that links the theoretical foundations of numerical analysis with computational applications for solving highly complex partial differential equations. Following the examination of the mathematical properties of these equations and the identification of the associated challenges—such as nonlinearity, multiscale behavior, numerical stability, and computational cost—the need arises for a systematic methodology that ensures the development of numerical algorithms capable of addressing these challenges with both efficiency and accuracy.In this research, the algorithm development methodology is based on integrating mathematical analysis with numerical experimentation, aiming to design a comprehensive algorithm that accounts for both the spatial and temporal aspects of the solution while achieving an effective

balance between accuracy, numerical stability, and computational efficiency. This approach includes the selection of appropriate spatial and temporal discretization methods suited to the nature of the equations under consideration, as well as the adoption of advanced strategies for handling nonlinearity and accelerating the convergence of numerical solutions.

Furthermore, the methodology is founded on the principle of incremental algorithm development, whereby an initial basic numerical model is first formulated and then progressively enhanced through the incorporation of mesh adaptivity techniques, error control mechanisms, and the use of preconditioners and multigrid methods. This staged approach allows for the independent assessment of the impact of each enhancement on the overall algorithmic performance, thereby contributing to the development of a more efficient and reliable numerical design.

### 3.1 Mathematical Formulation of the Selected Problems

The development of improved numerical algorithms for solving highly complex partial differential equations relies on the selection of a set of benchmark problems that represent different forms of numerical complexity, such as nonlinearity, multiscale behavior, variable coefficients, and the presence of boundary layers or shock waves. To ensure comparability and verifiability, these problems are formulated in a general mathematical framework that includes the spatial domain, the temporal interval, the governing equation, as well as the initial and boundary conditions.In general, the problem is defined on a spatial domai[ \Omega \subset \mathbb{R}^d, \quad d = 1, 2, 3,and over a time interval ((0, T]). The objective is to determine the function (u(x,t)) such that it satisfies the governing partial differential equation together with the prescribed initial and boundary conditions.

$$(0,T] \times \Omega \text{ في} f = \mathcal{L}(u)$$

With initial conditions:

$$\text{في } \Omega \;\; u_0(x) = \; u(x,0)$$

Boundary conditions on $\partial\Omega$are specified according to the nature of the problem, such as:

**Dirichlet condition:**

$u = g_D$on $\Gamma_D$

**Neumann condition:**

$\nabla u \cdot \mathbf{n} = g_N$on $\Gamma_N$

**Mixed (Robin) condition:**

$\alpha u + \beta \nabla u \cdot \mathbf{n} = g_R$

where $\mathbf{n}$denotes the outward unit normal vector, and

$\Gamma_D \cup \Gamma_N = \partial\Omega$.

$$(0,T] \times \Omega \ f(x,t)\text{ِ} = (k(x)(\nabla u) - \nabla \frac{\partial u}{\partial t}$$

This problem represents a fundamental model for capturing the complexity arising from medium heterogeneity and spatial variations in material properties. It can be formulated as follows:

where $k(\mathbf{x}) > 0$denotes the diffusion coefficient, which may be variable or spatially non-uniform. This problem is particularly useful for assessing the ability of numerical algorithms to handle sharp spatial variations in the coefficients, along with the associated challenges in accuracy and stability.

Advection–Diffusion Problem This equation poses a significant numerical challenge due to the possible formation of boundary layers and spurious oscillations when advection effects dominate the diffusion process (advection-dominated regime).

$$(0,T] \times \Omega \ \text{ِ}f(x,t) = (\varepsilon \nabla u) \ \cdot \nabla - \nabla u \cdot \mathrm{b}(x) \frac{\partial u}{\partial t} +$$

where $\mathbf{b}(\mathbf{x})$denotes the advection velocity vector, and $\varepsilon > 0$is the diffusion coefficient, which may be very small.

This problem is commonly used to evaluate the effectiveness of numerical stabilization techniques in suppressing non-physical oscillations, as well as to assess mesh-adaptivity strategies in regions containing boundary layers.

**2.3 Nonlinear Burgers' Equation**

The Burgers' equation is regarded as a canonical model for nonlinearity and shock formation at small viscosity values. It can be written (in one or more spatial dimensions) as follows:

$$(0,T] \in t(a,b) \in, f(x,t), x + \ \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}$$

where $\nu > 0$denotes the viscosity, which may be small. This equation is widely used to test the treatment of nonlinearity (e.g., Newton–Krylov methods) and the ability to resolve sharp fronts without loss of stability or accuracy.

**3.3 Wave Equation**

The wave equation represents an important model for oscillatory phenomena and wave propagation, exhibiting high sensitivity to the choice of the time-step size.

$$[T,0) \times \Omega \text{ في} \quad f(x,t) = c^2 \Delta u - \frac{u^2 \partial}{^2 t \partial}$$

With initial conditions:

$$v_0(x) = u_0(x), \qquad \frac{u\partial}{t\partial}(x,0) = u(x,0)$$

It is used to evaluate time-integration algorithms (such as Newmark or Runge–Kutta schemes) in terms of their ability to preserve wave characteristics (e.g., approximate energy balance) and to minimize numerical dispersion.

**4.3 Algorithm Design (Space / Time / Nonlinearity)**

The design of the proposed numerical algorithm aims to construct an integrated solution framework for highly complex partial differential equations that balances accuracy, stability, and computational efficiency. This is achieved through three tightly coupled layers: (1) spatial discretization, (2) time integration, and (3) nonlinear treatment, with systematic mechanisms defined for their mutual coupling.

1) Spatial Discretization Component

The spatial component is designed to accurately represent sharp gradients (boundary layers, fronts, and shocks), while accommodating complex geometries and spatially varying coefficients. Accordingly, the design adopts one of the following widely used approaches for complex PDEs:

Finite Element Methods (FEM) or High-Order Finite Elements (p-FEM / Spectral Elements): The problem is formulated in a weak form over the domain $\Omega$, and the solution is approximated within a space of local basis functions defined on the mesh elements. This approach offers significant geometric flexibility and enables accuracy enhancement through polynomial order elevation without a substantial increase in the number of elements (Hesthaven & Warburton, 2008, p. 87).

Discontinuous Galerkin (DG) or Finite Volume Methods (FVM) for Advection-Dominated Problems: When advection effects dominate or shock formation is expected, DG/FVM approaches are preferred due to their local conservation properties and their ability to incorporate numerical limiters or dissipative mechanisms that suppress non-physical oscillations, such as flux limiters or upwinding techniques.To ensure spatial stability in advection–diffusion problems, the algorithm incorporates one of the following stabilization strategies:

Upwind fluxes (in FVM/DG), or

SUPG / stabilized FEM formulations (in FEM).

The spatial discretization typically results in a semi-discrete algebraic system (semi-discrete in time):

$$\mathrm{f}(t) = \mathrm{R}(\mathrm{u}) + \frac{d\mathrm{u}}{dt} M$$

where $\mathbf{M}$denotes the mass matrix, and $\mathbf{R}$represents the residual vector arising from the spatial discretization.

**2) Time Integration Component**

The choice of the time-integration method depends on the nature of the problem, particularly its stiffness and multiscale characteristics. Accordingly, the proposed design adopts a flexible time-integration strategy based on one of the following paradigms:

**Implicit methods for stiff components:**

Methods such as Backward Euler or BDF2 are well suited when diffusion effects are strong, when $\varepsilon$is small, or when stability constraints in explicit schemes would require prohibitively small time steps.

**Explicit methods for non-stiff components:**

Explicit schemes, such as Runge–Kutta methods, are advantageous for reducing the computational cost associated with solving algebraic systems at each time step; however, they are restricted by the Courant–Friedrichs–Lewy (CFL) condition.

**IMEX (Implicit–Explicit) methods:**

These methods are particularly suitable for highly complex problems, as the governing equation is decomposed into a stiff part treated implicitly and a non-stiff part treated explicitly:

$$^{n+1}\mathbf{f} = \mathbf{R}_{\text{nonstiff}}(\mathbf{u}^n) + \mathbf{R}_{\text{stiff}}(\mathbf{u}^{n+1}) + \frac{^{n}\mathbf{u}^{n+1} - \mathbf{u}}{\Delta t} M$$

This hybrid integration allows for larger time steps than purely explicit schemes, while incurring a lower computational cost than fully implicit methods. In addition, adaptive time-step control is incorporated when necessary through temporal error indicators, particularly in cases where the solution dynamics vary rapidly (Hundsdorfer & Verwer, 2003, p. 156).

**3) Nonlinear Solver Strategy**

In many complex PDEs (such as the Burgers', Navier–Stokes, or reaction–diffusion equations), spatial and temporal discretization leads to a nonlinear system that must be solved at each time step:

$$0 = \mathrm{F}(\mathrm{u}^{n+1})$$

To solve this system efficiently, one of the following approaches is adopted:

**Newton's Method:**

The solution is updated iteratively according to:

$$^{(k)}\delta\mathbf{u} + {}^{(k)}\mathbf{u} = {}^{(k+1)}\mathbf{F}(\mathbf{u}^{(k)}), \quad \mathbf{u}- = {}^{(k)}J(\mathbf{u}^{(k)})\,\delta\mathbf{u}$$

where $\mathbf{J}$denotes the Jacobian matrix. Newton's method is characterized by quadratic convergence near the solution; however, it requires the construction or approximation of $\mathbf{J}$and the solution of large linear systems.

**Newton–Krylov methods (iterative linear solvers within Newton):** To reduce computational cost, the linear system arising at each Newton iteration is solved using iterative solvers such as GMRES or CG instead of direct methods, combined with effective preconditioners (e.g., ILU or AMG).

To enhance robustness for strongly nonlinear problems, the algorithm incorporates one of the following techniques:

- **Line search / damping**, to prevent divergence of Newton's iterations, or
- **Continuation methods** (parameter continuation), which gradually vary problem parameters to reach a stable solution.

**4) General Algorithmic Workflow (Concise)**

1- Generate the computational mesh and define the spatial approximation space, including the choice of element or control-volume type.

2- Assemble the residual vector $\mathbf{R}(\mathbf{u})$and system matrices (e.g., the mass matrix $\mathbf{M}$).

3- Select the time-integration scheme (implicit / IMEX) and determine the time-step size $\Delta t$.

4- At each time step:

- Formulate the resulting linear or nonlinear system,
- Solve the nonlinearity (Newton / Newton–Krylov),
- Update the solution and verify convergence criteria.

5- *(Optional)* Perform temporal and/or mesh adaptivity based on error indicators.

**4.3 Adaptivity Strategy (h / p / hp)**

Mesh adaptivity strategies constitute one of the most important modern tools for improving the efficiency and accuracy of numerical solutions of highly complex partial differential equations. Their

primary objective is to intelligently distribute computational resources by concentrating numerical effort in regions exhibiting high gradients or irregular behavior, while reducing cost in smooth regions. These strategies are generally classified into three main types: spatial adaptivity (h-adaptivity), polynomial-order adaptivity (p-adaptivity), and hybrid adaptivity (hp-adaptivity).

**Spatial Adaptivity (h-adaptivity)**

Spatial adaptivity is based on modifying the mesh element size $h$without changing the polynomial order of approximation within each element. Elements are refined in regions where the solution exhibits:

- Sharp gradients,
- Boundary layers,
- Discontinuities or shock-like behavior,

by subdividing them into smaller elements to enhance local accuracy.

**Advantages:**

- Highly effective for representing irregular solution behavior.
- Easily integrated with finite element (FEM) and finite volume (FVM) methods.
- Well suited for advection–diffusion problems with strongly varying coefficients.

**Challenges:**

- Rapid growth in the number of degrees of freedom.
- Increased complexity in mesh management and node renumbering.
- Difficulty in maintaining element quality for complex geometries.

**Polynomial-Order Adaptivity (p-adaptivity)**

This approach relies on increasing or decreasing the polynomial order $p$of the approximation functions within elements, while keeping the mesh fixed. It is most effective when the solution is:

- Smooth,
- Free of shocks or sharp discontinuities.

**Advantages:**

- Achieves high convergence rates (often exponential) for smooth problems.
- Does not require mesh regeneration.
- Lower storage cost compared to h-adaptivity at comparable accuracy levels.

**Challenges:**

- Ineffective in regions with strong irregularities.
- Increased complexity of the resulting algebraic system matrices.
- Higher sensitivity to numerical stability issues at large polynomial orders.

**Hybrid Adaptivity (hp-adaptivity)**

Hybrid hp-adaptivity represents one of the most advanced and effective strategies, combining:

- Element size reduction ($h$) in regions with complex solution behavior, and
- Polynomial order enrichment ($p$) in smooth regions.

This approach relies on intelligent error indicators capable of locally identifying the nature of the solution and determining whether optimal improvement is achieved through h-refinement, p-enrichment, or both.

**Advantages:**

- Provides the highest numerical efficiency among adaptivity strategies.
- Exhibits very fast (often exponential) convergence for many problems.
- Particularly well suited for multiscale and multiphysics PDEs.

**Challenges:**

- Increased algorithmic and data-management complexity.
- Difficulty in designing optimal decision criteria between h and p refinement.
- Requires advanced software infrastructure (Johnson, 2009, p. 64).

**Error Indicators and Decision-Making Mechanisms**

All adaptivity strategies rely on **a posteriori error estimation**, including:

- Residual-based indicators,
- Inter-element jump indicators,
- Energy-norm error indicators,
- Spectral-based indicators.

These indicators are used to determine:

- **Where** to adapt,
- Which type of adaptivity to apply (h, p, or both), and
- How much refinement or enrichment is required.

**Role of Adaptivity in Improving the Numerical Algorithm**

The adoption of h/p/hp adaptivity strategies contributes to:

- Improved accuracy without excessive increases in computational cost.
- Enhanced stability, particularly for stiff problems.
- Reduction of overall simulation runtime.
- Improved parallel efficiency through balanced workload distribution.

**5.3 Stability and Convergence Analysis (Theoretical, as Far as Possible)**

Stability and convergence analysis constitutes a cornerstone in the development of numerical algorithms for solving highly complex partial differential equations. It ensures that the numerical solution remains bounded over time and under mesh refinement, and that it converges to the exact solution as the time step is reduced or as the mesh is refined and/or the approximation order is increased.

The analysis is presented in a general framework applicable to a wide range of methods (FDM, FEM, FVM, and DG), focusing on the most commonly reported results in the literature: consistency, stability, and convergence theorems, as well as semi-discrete and fully discrete formulations (LeVeque, 2007, p. 118).

**1) Conceptual Framework: Consistency – Stability – Convergence**

Consistency

A numerical algorithm is said to be consistent if the local truncation error vanishes as $h \to 0$and/or $\Delta t \to 0$.

For spatial discretization methods: Consistency implies that the numerical approximation of the differential operator converges to the exact continuous operator.

For time-integration methods: Consistency means that the time-integration scheme approximates the time derivative with an accuracy of order $O(\Delta t^{\,q})$, where $q$denotes the temporal order of accuracy.

Stability

Stability refers to the property that errors arising from initial data, discretization, or numerical approximation do not grow unbounded during time integration or while solving the resulting algebraic system. It is often expressed in the form of an upper bound of the type:

$$T \geq n\Delta t \geq 0 \quad ,\left((\text{مصادر خطأ أخرى}) + \|^{0}C\left(\|e \geq \|^{n}e\right)\right.$$

where $C$is a constant that does not blow up with respect to the time level $n$or under mesh and time-step refinement $h, \Delta t$, provided certain conditions are satisfied.

**Convergence**

Convergence means that:

$$0 \longrightarrow h, \Delta t \text{ عندما } \quad 0 \longrightarrow \| {}_h^n u - u(\cdot, t_n) \|$$

The order of convergence is often derived by combining **consistency** with **stability**

$$\left( {}^q \Delta t + {}^p C \left( h \geq \| {}_h^n u - u(\cdot, t_n) \| \right. \right.$$

where $p$denotes the order of spatial approximation and $q$denotes the order of temporal approximation.

**Methodological remark:**

For linear equations, the Lax–Richtmyer theorem (for well-posed initial value problems) is commonly employed, stating that consistency + stability implies convergence. In contrast, for nonlinear problems, one typically relies on Lipschitz stability or energy stability, together with appropriate residual estimates.

2) Temporal Stability Analysis: Absolute Stability Perspective

When applying a time-integration scheme to the test equation:

$$0 \geq \lambda y, \qquad \Re(\lambda) = {}' y$$

The amplification factor $R(z)$is defined, where $z = \lambda \Delta t$. A time-integration scheme is considered stable if

$$| R(z) | \leq 1 \text{for } z \text{ within the appropriate stability region.}$$

Explicit schemes are typically conditionally stable and require a Courant–Friedrichs–Lewy (CFL)–type constraint.

Implicit schemes may be A-stable (stable for all $\mathrm{Re}\,(z) \leq 0$), and some are L-stable, meaning that they effectively damp strong stiffness effects. Such schemes are well suited for stiff diffusion or reaction problems.

IMEX schemes treat the stiff components implicitly and the non-stiff components explicitly, thereby achieving a compromise between stability and computational cost.

General Formulation of the CFL Condition (for Advection–Diffusion Problems):

$$(\frac{h}{|\,a\,|},\ \frac{h^2}{\nu})\ C\ \min\ \leq \Delta t$$

**Spatial Stability: Energy Stability**

For many classes of PDEs (such as heat/diffusion equations and linear elasticity), it is possible to construct an **energy functional** $E(t)$that satisfies

$$0 \leq E(t)\frac{d}{dt}$$

for the continuous system. The numerical objective is to preserve an analogous property at the discrete level:

$$E^n \leq E^{n+1} + (\text{"Controlled boundaries"})$$

**Canonical example (diffusion/heat):**

For the equation

$$u_t = \kappa\Delta u + f,$$

(or equivalently $f = \kappa\Delta u - u_t$), under appropriate boundary conditions, one can prove that:

Energy Estimate (Heat/Diffusion)

For

$$f(s)\|_{2}^{2}{}_{L}ds\|_{0}^{t}\int + {}_{2}^{2}{}_{L}\|u(0)\| \geq 2\kappa\int_{0}^{t}\|\nabla u(s)\|_{2}^{2}{}_{L}ds + {}_{2}^{2}{}_{L}\|u(t)\|$$

Numerically, the objective is to obtain a similar estimate by exploiting the symmetry of the stiffness matrix (in FEM) or the conservation/dissipation properties in FVM/DG, combined with appropriate stabilization.

**Convection-Dominated Problems**

For convection-dominated problems, numerical oscillations may arise in the absence of stabilization. Therefore, techniques such as:

**Upwinding**, **SUPG**, or **limiters**,

are employed to achieve energy or generalized stability and to suppress non-physical oscillations.

**Convergence Analysis: Error Decomposition and Estimates**

The numerical error is typically analyzed by decomposing it into:

Projection / spatial approximation error,

Temporal discretization error,

Nonlinear or algebraic solver error (when applicable).

Semi-Discrete (Spatially Discrete) Analysis

If the exact solution $u$possesses sufficient regularity, then finite element methods, for example, yield:

$$\|C\, h^{p+1}\|_{H^{p+1}} \|u(t)\|_{L^2} \geq \|u_h(t) - u(t)\|$$

$$\|C\, h^{p}\|_{H^{p+1}} \|u(t)\|_{L^2} \geq \|(u_h(t) - u(t))\nabla\|$$

**Fully Discrete Error Decomposition**

If the time-integration scheme is of order $q$, then the total error typically satisfies:

$$\| u(t_n) - u_h^n \| \leq C\,(h^p + \Delta t^q)$$

**Fully Discrete Error Decomposition**

If the time-integration scheme is of order $q$, the total error typically satisfies the estimate

$$\| u(t_n) - u_h^n \| \leq C(h^p + \Delta t^q),$$

where $C$is a constant independent of $h$and $\Delta t$, provided that the exact solution possesses sufficient regularity.

$$\left(\Delta t^{q} + C^{p+1}\left(h \geq \|u_h^n - u(t_n)\|\right.\right.$$

with a constant $C$that depends on the final simulation time $T$and on the regularity properties of the exact solution. Preferably, this constant should be insensitive to stiffness when an implicit or L-stable time-integration scheme is employed.

Stability and Convergence in Nonlinear Problems (Nonlinear PDEs)

For nonlinear problems (such as the Navier–Stokes equations or nonlinear reaction diffusion systems), analytical tools commonly employed include:Local Lipschitz continuity conditions on the nonlinear operator:

$$\|v - L\|u \geq \|N(v) - N(u)\|$$

or energy stability (via skew-symmetry or monotonicity properties) to control solution growth.

Together with Gronwall's inequality, bounds of the form

$$\| e^n \| \leq \exp(CT) \text{ (consistency errors)}$$

can be derived. This highlights the importance of selecting spatial and temporal schemes that limit the amplification of constants and ensure long-time stability.

6) Effect of Adaptivity (h / p / hp) on Stability and Convergence

- h-adaptivity:

  Improves local accuracy but may impose more restrictive time-step constraints (CFL condition) when explicit schemes are used.

  p-adaptivity:

  Increases the order of convergence in smooth regions, but may exacerbate stability sensitivity in advection-dominated regimes unless stabilization is employed (Quarteroni et al., 2007, p. 203).

  hp-adaptivity:

  Can achieve superior convergence provided that:

  - a reliable error estimator is available,
  - an appropriate decision rule (when to apply h- or p-refinement) is adopted, and
  - mesh quality is preserved, avoiding highly distorted elements.

7) Methodological Commitments of This Research

To make the stability and convergence analysis a practical component of the dissertation, the following commitments are adopted:

- Prove consistency of the proposed scheme, with spatial order $p$and temporal order $q$.
- Perform linear stability analysis via:
  - Von Neumann analysis (when applicable), or
  - Energy estimates for diffusion/dissipative operators.
- Derive a CFL condition for the explicit component (if present) and clarify how IMEX or implicit schemes alleviate stability restrictions.
- Establish a convergence proof for at least one canonical linear problem, and provide a semi-theoretical justification for nonlinear problems using Lipschitz/energy stability combined with Gronwall's inequality.
- Support the analysis with experimental order of convergence (EOC) studies demonstrating:
  - the expected convergence rates,
  - long-time stability behavior, and
  - the impact of preconditioners and multigrid techniques on the stability of the algebraic solvers.

Test Suite and Benchmarks

This section aims to construct a standardized test suite for fair and reproducible evaluation of the proposed algorithm, comparing accuracy, stability, computational cost, and parallel scalability on both classical and challenging PDE benchmarks (Thomée, 2006, p. 97).

Objectives of the Tests

- Verification: Does the numerical solution converge to a known exact solution or a high-fidelity reference solution?
- Order verification: Are the expected spatial order $p$and temporal order $q$achieved?
- Stability assessment: Does the solution remain bounded without spurious oscillations or numerical blow-up over long time intervals?
- Efficiency measurement: Runtime, iteration counts, and memory usage.
- Robustness evaluation: Performance under heterogeneous coefficients, irregular meshes, and varied boundary conditions.
- Adaptivity assessment (h/p/hp): Does the method concentrate computational resources where needed while reducing degrees of freedom without sacrificing accuracy?

**2) Selection of Representative Benchmarks**

(a) Problems with Analytical or Manufactured Solutions

Used to calibrate accuracy and measure convergence orders precisely:

1- Poisson equation:

$f = \Delta u$ −on a square or cube with Dirichlet or Neumann boundary conditions.

Heat equation:

$u_t - \kappa \Delta u = f$(testing temporal stability and convergence orders).

2-Pure advection: (testing oscillations and CFL constraints).

The Method of Manufactured Solutions (MMS) is preferably employed to construct an exact solution $u$and derive the corresponding source term $f$, even for complex problem settings (Trefethen, 2000, p. 22).

(b) Stress-Test Problems Representing High Complexity

- Advection–diffusion (convection-dominated): Large Péclet number $\mathrm{Pe} \gg 1$, to assess stabilization techniques (upwinding, SUPG, limiters).
- Reaction–diffusion (stiffness-dominated): To test IMEX or L-stable implicit schemes.

- Burgers' / nonlinear convection–diffusion equations: To evaluate nonlinear solvers and Newton–Krylov strategies.
- Incompressible Navier–Stokes (2D lid-driven cavity): A stringent benchmark for stability, convergence, and mesh sensitivity.

(c) Application-Oriented Use Case

- Selection of one realistic application relevant to the specialization (e.g., flow, heat transfer, contaminant transport), with reasonable data and boundary conditions, to demonstrate the practical effectiveness of the proposed algorithm.

**3) Definition of Test Conditions (Domains / BCs / ICs)**

For each problem, the following are specified precisely:

- Domain: Ω(2D/3D) and its geometry (square, channel, cavity, etc.).
- Boundary conditions: Dirichlet, Neumann, Robin, or periodic.
- Initial conditions (for time-dependent problems).
- Model parameters: $\kappa, \nu, a$, and their spatial variability (to induce complexity).
- Initial mesh and refinement strategies: uniform or non-uniform.

4) Evaluation Metrics and Criteria

$$\|e\|_{L^2}, \quad \|e\|_{H^1}, \quad \|e\|_{L^\infty}$$

- خطأ نسبي:

$$\mathrm{RelErr} = \frac{\|u_h - u\|}{\|u\|}$$

(ب) رتبة التقارب (EOC)
باستخدام تقليل $h$ أو $\Delta t$:

$$\mathrm{EOC} = \frac{\log(\|e_1\| / \|e_2\|)}{\log(h_1 / h_2)}$$

وبالمثل للزمن:

$$\mathrm{EOC}_t = \frac{\log(\|e_{\Delta t_1}\| / \|e_{\Delta t_2}\|)}{\log(\Delta t_1 / \Delta t_2)}$$

**(c) Stability**

**Monitoring the growth or decay of an energy norm or the $L^2$-norm:**

$$E^n = \|u_h^n\|_{L^2}^2 \quad \text{أو} \quad E^n = \|\nabla u_h^n\|^2 + \alpha \|u_h^n\|^2$$

**(d) Computational Efficiency**

- Total execution time (wall-clock time).
- Number of iterations for linear and nonlinear solvers (GMRES / Newton iterations).
- Cost per time step (Trottenberg et al., 2001, p. 311).
- Memory consumption.
- Adaptivity overhead: number of refinement/coarsening operations.

(e) Effectiveness of Preconditioners and Multigrid

- Krylov convergence rate: fewer iterations for the same tolerance.
- Performance sensitivity with respect to $h$(grid-independent convergence, if achievable).

**(f) Parallel Scalability**

Strong scaling: fixed problem size with increasing number of processors.

Weak scaling: problem size increased proportionally with the number of processors.

Parallel efficiency:

$$\eta = \frac{T_1}{p\,T_p},$$

where $T_1$is the runtime on one processor, $T_p$is the runtime on $p$processors.

5) Fair Algorithm Comparison Protocol

To ensure a fair comparison between the proposed method and reference methods, the following guidelines are adopted:

- Use identical domains, boundary conditions, and model parameters.
- Enforce uniform stopping criteria for algebraic solvers:
  - GMRES tolerance, e.g., $10^{-8}$,
  - Newton tolerance, e.g., $10^{-10}$.
- Standardize the target accuracy level (error target) rather than fixing the mesh alone.
- Report accuracy versus computational cost (error vs. CPU time), not accuracy alone.

**Adaptivity-Specific Tests (h / p / hp)**

- Compare three strategies: h-only, p-only, and hp adaptivity.
- Additional metrics:
  - Number of degrees of freedom (DOFs) required to reach the same error level.
  - Distribution of mesh elements and polynomial orders (refinement maps).

- Adaptivity cost versus achieved accuracy gains.

Clear success criterion:

hp-adaptivity achieves the same error with fewer DOFs or lower runtime compared to the other strategies.

6) Deliverables

Tables: error values, EOC, runtime, iteration counts, and memory usage.

Plots:

Error vs. $h$,

Error vs. CPU time,

Iterations vs. DOFs.

Visualizations: mesh configurations and polynomial-order maps illustrating the effect of adaptivity.

Executive summary for each benchmark: what is demonstrated, and the key strengths and limitations.

**Conclusions**

1- Highly complex partial differential equations (PDEs) constitute a genuine numerical challenge that cannot be efficiently addressed using traditional algorithms without introducing substantial improvements.

2- The analysis demonstrates that numerical stability is a fundamental requirement for ensuring solution convergence, particularly in nonlinear and multiscale problems.

3- The use of advanced spatial and temporal discretization methods—especially implicit and IMEX schemes—significantly enhances stability and alleviates severe time-step restrictions.

4- Mesh adaptivity strategies (h-, p-, and hp-adaptivity) have proven effective in improving accuracy while reducing the number of degrees of freedom, particularly in regions characterized by sharp gradients or boundary layers.

5- The integration of preconditioning techniques and multigrid methods with iterative solvers leads to substantial acceleration of convergence and a significant reduction in computational time.

6- Benchmark test results indicate that the improved numerical algorithms outperform conventional methods in terms of accuracy and computational efficiency at the same error level.

7- Parallel scalability has emerged as a critical factor in the design of modern numerical algorithms, especially for large-scale problems.

**Recommendations**

1- This research recommends adopting the proposed improved numerical algorithms in engineering and scientific applications that require high accuracy with low computational cost.
2- Further emphasis should be placed on the use of hybrid adaptivity strategies (hp-adaptivity) due to their strong capability to achieve superior convergence rates in complex problems.
3- The integration of multigrid methods as preconditioners within Krylov subspace solvers should be encouraged to achieve better computational performance for large algebraic systems.
4- Future research should be extended to address more complex classes of PDEs, such as multiphysics problems or systems coupled with stochastic effects.
5- Greater exploitation of parallel computing and modern hardware architectures (multi-core CPUs and GPUs) is recommended for the efficient implementation of numerical algorithms.
6- Additional comparative studies between the proposed algorithms and recently published state-of-the-art methods in the international literature are recommended to further validate the results.
7- The development of flexible and scalable software frameworks is recommended to facilitate the deployment of these algorithms in diverse research and industrial environments.

**References**


1- Al-Bayati, M. A. (2017). *Partial Differential Equations: Theory and Applications* (1st ed.). Dar Al-Fikr Al-Arabi.
2- Al-Jubouri, A. K. (2015). *Numerical Analysis and Its Engineering Applications* (1st ed.). Dar Al-Kutub Al-Ilmiyah.
3- Hussein, A. K. H. (2014). *Numerical Analysis* (2nd ed.). Dar Al-Masirah for Publishing and Distribution.
4- Al-Khafaji, A. H. (2013). *Numerical Methods in Engineering* (1st ed.). Dar Al-Masirah for Publishing and Distribution.
5- Al-Rawi, A. A. R. (2016). *Partial Differential Equations and Their Applications* (1st ed.). Dar Al-Yazouri Scientific Publishing.
6- Al-Samarrai, Q. M. (2012). *Advanced Numerical Methods* (1st ed.). Dar Safa for Publishing and Distribution.
7- Abdullah, F. H. (2018). *Numerical Solutions of Differential Equations* (2nd ed.). Dar Al-Manahij for Publishing and Distribution.

**English References**

8-Brenner, S. C., & Scott, R. (2008). *The mathematical theory of finite element methods* (3rd ed.). Springer.
https://doi.org/10.1007/978-0-387-75934-0

9-Cockburn, B., Karniadakis, G. E., & Shu, C.-W. (Eds.). (2000). *Discontinuous Galerkin methods: Theory, computation and applications*. Springer.
https://doi.org/10.1007/978-3-642-59721-3

10-Evans, L. C. (2010). *Partial differential equations* (2nd ed.). American Mathematical Society.
https://doi.org/10.1090/gsm/019

11-Hesthaven, J. S., & Warburton, T. (2008). *Nodal discontinuous Galerkin methods*. Springer.
https://doi.org/10.1007/978-0-387-72067-8

12-Hundsdorfer, W., & Verwer, J. G. (2003). *Numerical solution of time-dependent advection–diffusion–reaction equations*. Springer.
https://doi.org/10.1007/978-3-662-09017-6

13-Johnson, C. (2009). *Numerical solution of partial differential equations by the finite element method*. Dover Publications.

14-LeVeque, R. J. (2007). *Finite difference methods for ordinary and partial differential equations*. SIAM.
https://doi.org/10.1137/1.9780898717839

15-Quarteroni, A., Sacco, R., & Saleri, F. (2007). *Numerical mathematics* (2nd ed.). Springer.
https://doi.org/10.1007/978-3-540-34611-8

16-Saad, Y. (2003). *Iterative methods for sparse linear systems* (2nd ed.). SIAM.
https://doi.org/10.1137/1.9780898718003

17-Thomée, V. (2006). *Galerkin finite element methods for parabolic problems* (2nd ed.). Springer.
https://doi.org/10.1007/978-3-540-33121-3

18-Trefethen, L. N. (2000). *Spectral methods in MATLAB*. SIAM.
https://doi.org/10.1137/1.9780898719598

19-Trottenberg, U., Oosterlee, C. W., & Schüller, A. (2001). *Multigrid*. Academic Press.